JSP - EXCEPTION HANDLING

When you are writing JSP code, a programmer may leave a coding errors which can occur at any part of the code. You can have following type of errors in your JSP code:

- **Checked exceptions:** Achecked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.
- **Runtime exceptions:** A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compliation.
- **Errors:** These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

This tutorial will give you few simple and elegant ways to handle run time exception/error occuring in your JSP code.

Using Exception Object:

The exception object is an instance of a subclass of Throwable (e.g., java.lang. NullPointerException) and is only available in error pages. Following is the list of important medthods available in the Throwable class.

SN	Methods with Description
1	**public String getMessage()** Returns a detailed message about the exception that has occurred. This message is initialized in the Throwable constructor.
2	**public Throwable getCause()** Returns the cause of the exception as represented by a Throwable object.
3	**public String toString()** Returns the name of the class concatenated with the result of getMessage()
4	**public void printStackTrace()** Prints the result of toString() along with the stack trace to System.err, the error output stream.
5	**public StackTraceElement [] getStackTrace()** Returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack, and the last element in the array represents the method at the bottom of the call stack.
6	**public Throwable fillInStackTrace()** Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace.

JSP gives you an option to specify Error Page for each JSP. Whenever the page throws an exception, the JSP container automatically invokes the error page.

Following is an example to specifiy an error page for a main.jsp. To set up an error page, use the <%@ page

errorPage="xxx" %> directive.

```
<%@ page errorPage="ShowError.jsp" %>

<html>
<head>
   <title>Error Handling Example</title>
</head>
<body>
<%
   // Throw an exception to invoke the error page
   int x = 1;
   if (x == 1)
   {
      throw new RuntimeException("Error condition!!!");
   }
%>
</body>
</html>
```

Now you would have to write one Error Handling JSP ShowError.jsp, which is given below. Notice that the error-handling page includes the directive <%@ page isErrorPage="true" %>. This directive causes the JSP compiler to generate the exception instance variable.

```
<%@ page isErrorPage="true" %>
<html>
<head>
<title>Show Error Page</title>
</head>
<body>
<h1>Opps...</h1>
<p>Sorry, an error occurred.</p>
<p>Here is the exception stack trace: </p>
<pre>
<% exception.printStackTrace(response.getWriter()); %>
</pre>
</body>
</html>
```

Now try to access main.jsp, it should generate something as follows:

```
java.lang.RuntimeException: Error condition!!!
......

Opps...
Sorry, an error occurred.

Here is the exception stack trace:
```

Using JSTL tags for Error Page:

You can make use of JSTL tags to write an error page ShowError.jsp. This page has almost same logic which we have used in above example, but it has better structure and it provides more information:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page isErrorPage="true" %>
<html>
<head>
<title>Show Error Page</title>
</head>
<body>
<h1>Opps...</h1>
<table width="100%" border="1">
<tr valign="top">
<td width="40%"><b>Error:</b></td>
<td>${pageContext.exception}</td>
```

```
</tr>
<tr valign="top">
<td><b>URI:</b></td>
<td>${pageContext.errorData.requestURI}</td>
</tr>
<tr valign="top">
<td><b>Status code:</b></td>
<td>${pageContext.errorData.statusCode}</td>
</tr>
<tr valign="top">
<td><b>Stack trace:</b></td>
<td>
<c:forEach var="trace"
         items="${pageContext.exception.stackTrace}">
<p>${trace}</p>
</c:forEach>
</td>
</tr>
</table>
</body>
</html>
```

Now try to access main.jsp, it should generate something as follows:

OPPS...

Error:	java.lang.RuntimeException: Error condition!!!
URI:	/main.jsp
Status code:	500
Stack trace:	org.apache.jsp.main_jsp._jspService(main_jsp.java:65) org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:68) javax.servlet.http.HttpServlet.service(HttpServlet.java:722) org.apache.jasper.servlet.JspServlet.service(JspServlet.java:265) javax.servlet.http.HttpServlet.service(HttpServlet.java:722) .....

Using Try...Catch Block:

If you want to handle errors with in the same page and want to take some action instead of firing an error page, you can make use of **try....catch** block.

Following is a simple example which shows how to use try...catch block. Let us put following code in main.jsp:

```
<html>
<head>
   <title>Try...Catch Example</title>
</head>
<body>
```

```
<%
   try{
      int i = 1;
      i = i / 0;
      out.println("The answer is " + i);
   }
   catch (Exception e){
      out.println("An exception occurred: " + e.getMessage());
   }
%>
</body>
</html>
```

Now try to access main.jsp, it should generate something as follows:

```
An exception occurred: / by zero
```